IPS — Paměť a efektivita programu

Petr Peringer
peringer AT fit.vutbr.cz

Vysoké učení technické v Brně,
Fakulta informačních technologií,
Božetěchova 2,
61266 Brno

(Verze: 1. prosince 2022)

Základní pojmy

Obsah:

- Motivace, příklad
- Paměť (adresa, bajt, slovo/*word*)
- Přístupová doba (latence), přenosová rychlost (*bandwidth*)
- Paměťová hierarchie (registry, cache L1, L2, L3, RAM)
- Cache L1, velikost, asociativita
- Lokalita odkazů
- Vliv na výkon programů

Příklady viz `http://www.fit.vutbr.cz/~peringer/UNOFFICIAL/IPS/`

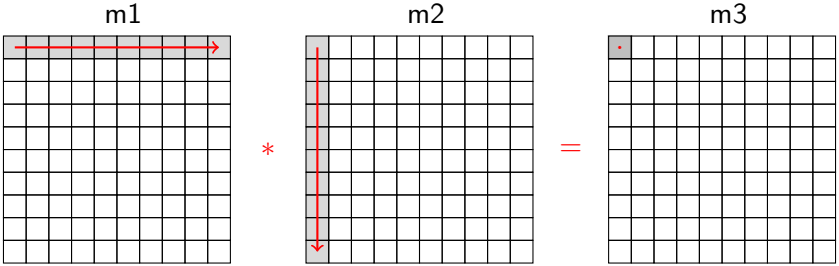
Drepper: *What Every Programmer Should Know About Memory*, 2007

Příklad1: násobení matic

Školní algoritmus násobení matic v C99, složitost $O(N^3)$:

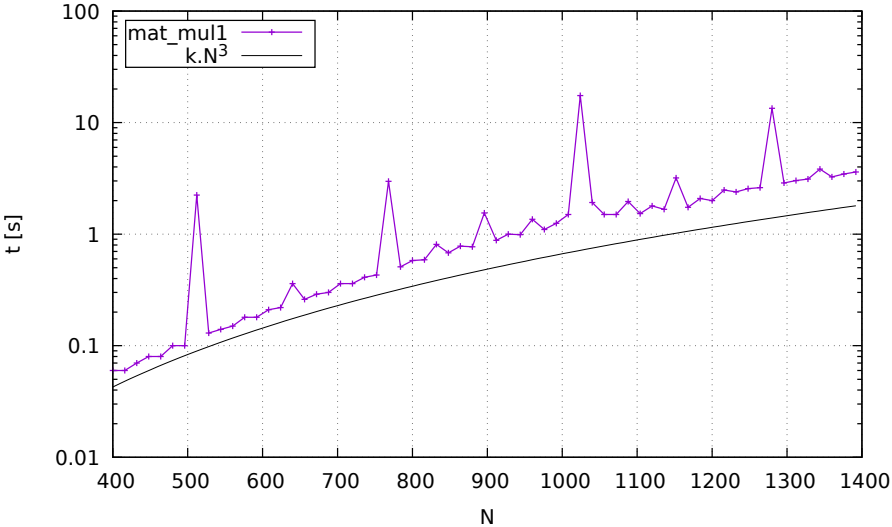
```
void
mat_mul1(int n, double m1[n][n], double m2[n][n], double m3[n][n])
{
    int i, j, k;
    memset(m3, 0, sizeof(double)*n*n);
    for (i = 0; i < n; ++i) {
        for (j = 0; j < n; ++j) {
            for (k = 0; k < n; ++k) {
                m3[i][j] += m1[i][k] * m2[k][j];
            }
        }
    }
}
```

Příklad1: násobení matic — školní algoritmus



Poznámka: C99 ukládá matice po řádcích (*row-major*)

Příklad1: násobení matic — výsledky (AMD Ryzen 3 2200G)

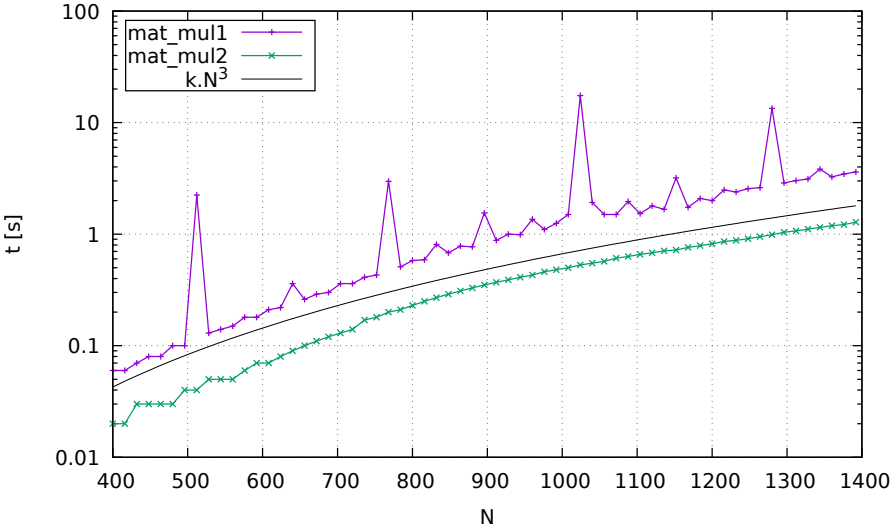


Příklad2: násobení matic

Vylepšený školní algoritmus (lepší lokalita odkazů):

```
void
mat_mul2(int n, double m1[n][n], double m2[n][n], double m3[n][n])
{
    int i, j, k;
    memset(m3, 0, sizeof(double)*n*n);
    for (i = 0; i < n; ++i) {
        for (k = 0; k < n; ++k) {       // změna: přehození k,j
            for (j = 0; j < n; ++j) {   //
                m3[i][j] += m1[i][k] * m2[k][j];
            }
        }
    }
}
```

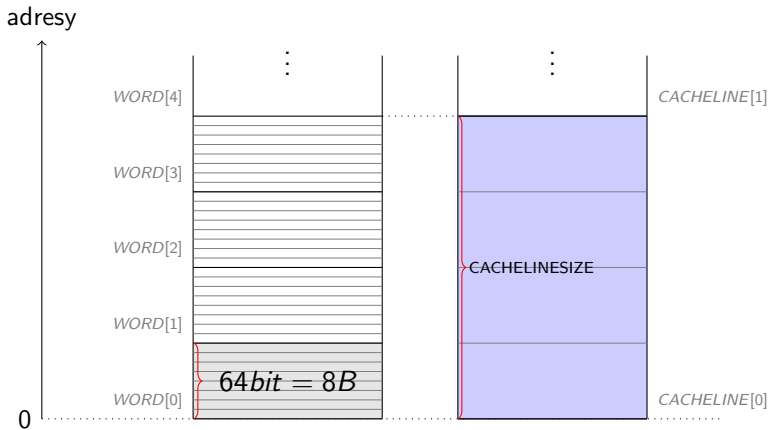
Příklad2: násobení matic — výsledky



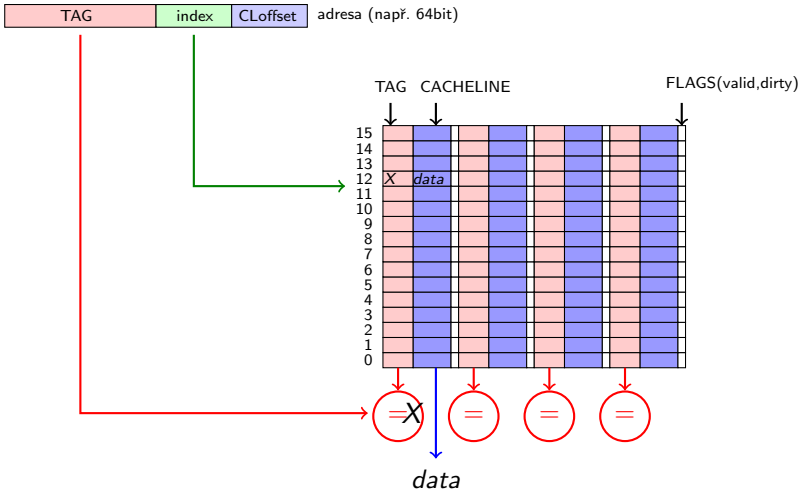
Lokalita odkazů

- Rozložení odkazů do paměti v čase/prostoru
- Souvislost s architekturou paměť. podsystému
- Vliv velikosti vyrovnávacích pamětí (především L1), pojmy: *cache hit/miss*, *hit rate*
- Problémy pro některé algoritmy
  - nelineární přístup (C matice po sloupcích)
  - náhodný přístup (hashing, seznamy)
  - ...

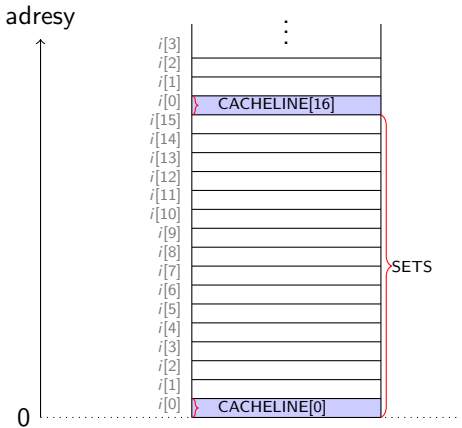
Hlavní paměť a cache



Cache L1, 4-way set associative, cache hit



Množinově asociativní cache



Konfigurace vyrovnávacích pamětí (viz příkaz `getconf`)

	Ryzen3 2200G	Core i5-4690
LEVEL1_ICACHE_SIZE	65536	32768
LEVEL1_ICACHE_ASSOC	4	8
LEVEL1_ICACHE_LINESIZE	64	64
LEVEL1_DCACHE_SIZE	32768	32768
LEVEL1_DCACHE_ASSOC	8	8
LEVEL1_DCACHE_LINESIZE	64	64
LEVEL2_CACHE_SIZE	524288	262144
LEVEL2_CACHE_ASSOC	8	8
LEVEL2_CACHE_LINESIZE	64	64
LEVEL3_CACHE_SIZE	4194304	6291456
LEVEL3_CACHE_ASSOC	16	12
LEVEL3_CACHE_LINESIZE	64	64

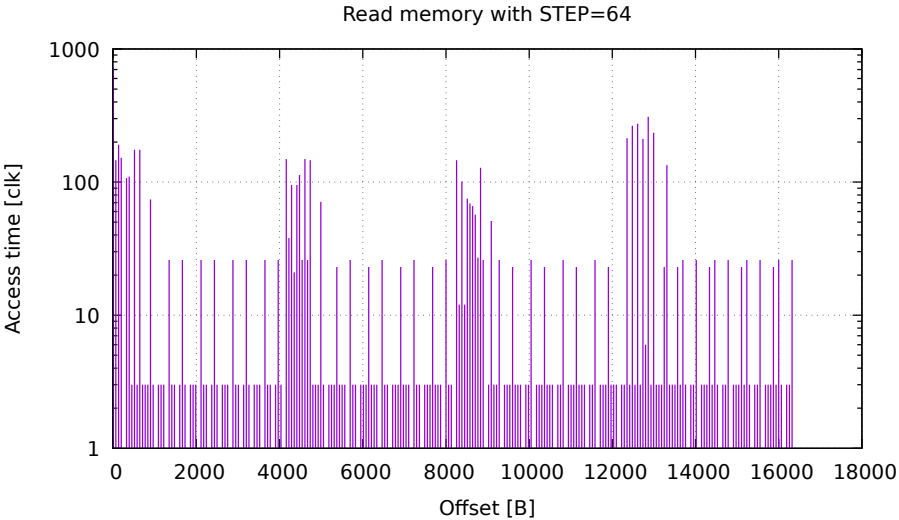
Doba přístupu

Pro *cache hit* je přístupová doba (latence):

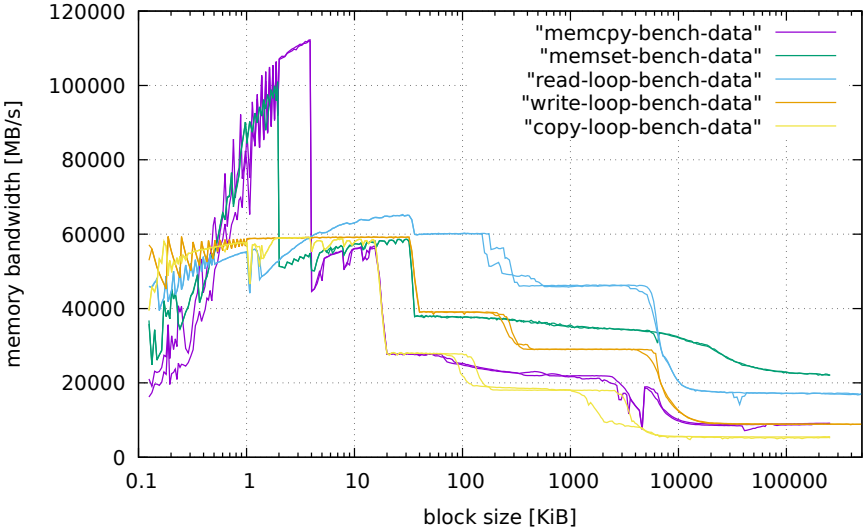
typ paměti	typická latence [takty CPU]
registry CPU	≤ 1
L1	3-4
L2	10
L3	40-60
DRAM	řádově stovky

Poznámky:
hit rate, *prefetch*, *write buffer*,
souvislost s VM: TLB, fyzický TAG, virtuální index, ...

Experiment: Doba přístupu (Intel i5)



Experiment: Přenosová rychlost (Intel i5)



Příklad — násobení matic, různé algoritmy

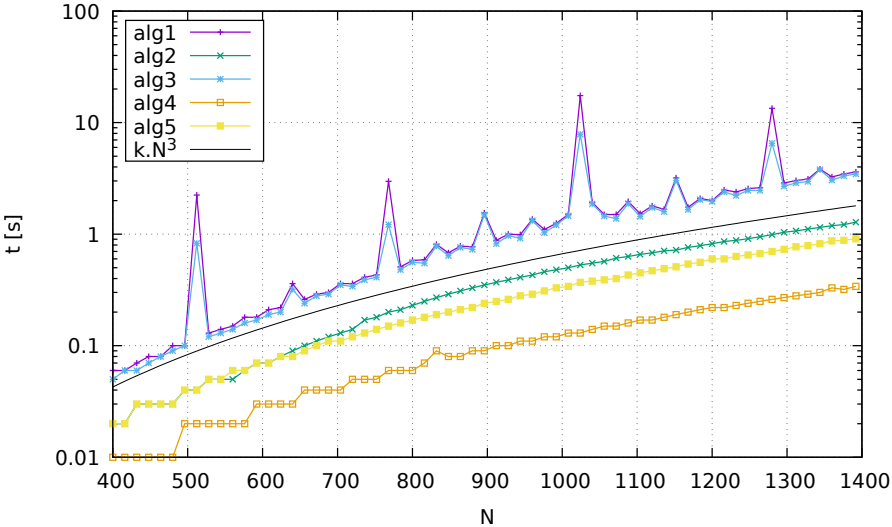
Experimentální výsledky pro procesory:

- *AMD Ryzen 3 2200G*
- *Intel Core i5-4690*

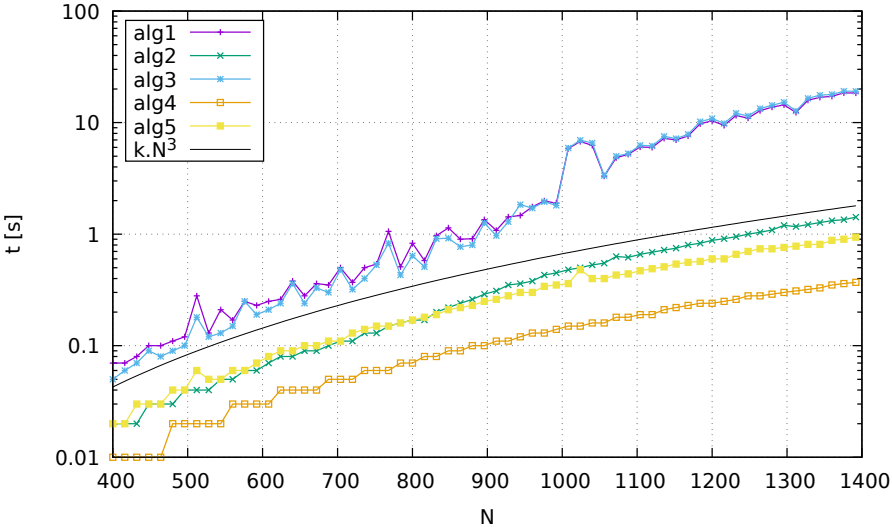
Algoritmy:

1. Školní
2. Školní s lepší lokalitou odkazů
3. Školní bez memset
4. Knihovna BLAS (cblas)
5. Násobení po blocích 64x64

Experiment: násobení matic (Ryzen3)



Experiment: násobení matic (i5)



Závěr

Shrnutí:

- Základní principy
- Zásadní vliv konfigurace HW na výsledky
- Lokalita odkazů a jak ji zlepšit
- Výhody použití standardních numerických knihoven
- Doplňková literatura viz WWW (Drepper2007)
- Souvislosti
- Co jsme úmyslně vynechali (cache coherence, prefetching, ...)
- Pokračování: nástroje cachegrind, perf, ...